

## APPENDIX A. REPRODUCIBILITY CHECKLIST (BORROWED FROM NLP FIELD)

### For all reported experimental results:

- A clear description of the mathematical setting, algorithm, and/or model  
: See in Section 3.
- Submission of a zip file containing source code, with specification of all dependencies, including external libraries, or a link to such resources (while still anonymized)  
: An anonymized Github link is attached.
- Description of computing infrastructure used  
: We used 6 Titan Xp (12G)
- Average runtime for each approach  
: The runtime depends on the number of processes running in the computing server. Approximately, it took 3 hour/epoch in large datasets (DBpedia, YelpReview) and 30min/epoch in small datasets. (YahooAnswer(Up&Low), AGNews, IMDB)
- Number of parameters in each model : A TextCNN model has 136K parameters except for word embedding parameters. It varies by the size of word embeddings.
- Corresponding validation performance for each reported test result  
: See in Figure 2. We will do the experiments again and report the result in rebuttal period if reviewers need the validation performance in the baselines.
- Explanation of evaluation metrics used, with links to code  
: All the metric on the classification tasks was accuracy.

### For all experiments with hyperparameter search:

- Bounds for each hyperparameter
- Hyperparameter configurations for best-performing models
- Number of hyperparameter search trials
- The method of choosing hyperparameter values (e.g., uniform sampling, manual tuning, etc.) and the criterion used to select among them (e.g., accuracy)
- Expected validation performance, or the mean and variance as a function of the number of hyperparameter trials

: To these all bullets, since our approach was tested in 6 different text classification tasks, we did not search specific hyperparameters. Instead, we followed general hyperparameters widely used in other studies. (e.g., Adam optimizer, 1e-3 learning rate, kernel size, multi-channel approach)

### For all datasets used:

- Relevant statistics such as number of examples  
: See Table 1.
- Details of train/validation/test splits  
: See Table 1 and Section 4.1.
- Explanation of any data that were excluded, and all pre-processing steps  
: No data were excluded. In the pre-processing step, we added space before and after special symbols (e.g., !@\$%) and then we tokenized the raw text by space.
- A link to a downloadable version of the data  
: [https://drive.google.com/drive/u/0/folders/0Bz8a\\_Dbh9Qhbfl16bVpmNUtUcFdjYmF2SEpmZUZUCvNiMUw1TWN6RDV3a0JHT3kxLVhVR2M](https://drive.google.com/drive/u/0/folders/0Bz8a_Dbh9Qhbfl16bVpmNUtUcFdjYmF2SEpmZUZUCvNiMUw1TWN6RDV3a0JHT3kxLVhVR2M)

When you copy and paste this link, make sure that ‘\_’ is included in your copied text \*\*  
Because of some bugs, the above link is oversized when we wrap the link using url  
: for YahooAnswer, we used [https://cogcomp.seas.upenn.edu/page/resource\\_](https://cogcomp.seas.upenn.edu/page/resource_)

view/89

- For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control
- : No new data were collected

## APPENDIX B. DATA STATISTICS

Table 9: The data information used in text classification. YahooAnswer dataset is used for two different tasks, which are to classify upper-level categories and to classify lower-level categories, respectively. The vocabulary size can be slightly different due to the tokenizing methods and the predefined special tokens.

	<b>DBpedia</b>	<b>Yah(Up)</b>	<b>Yah(Low)</b>	<b>AGNews</b>	<b>Yelp</b>	<b>IMDB</b>
#Train	560,000	133,703	133,703	120,000	650,000	25,000
#Test	70,000	23,595	23,595	7,600	50,000	25,000
#Class	14	17	280	4	5	2
#Vocab	626,717	154,142	154,142	66,049	198,625	47,113

## APPENDIX C. MODEL COMPARISON &amp; JUSTIFICATION

First, this is the performance of Very Deep Convolutional Neural Net (Conneau et al., 2017) according to their report, github implementation, and our modification. Note again that YahooAnswer dataset is different, but it is the original dataset, as described in Appendix.

Table 10: The performance of TextCNN classifiers and the standard deviation with GraVeR according to the gradualness policy. The default method is to increase the number of maskers when the validation performance decreases.

	<b>DBpedia</b>	<b>Yah(Up)</b>	<b>Yah(Low)</b>	<b>AGNews</b>	<b>Yelp</b>	<b>IMDB</b>
TextCNN <sub>base</sub> (Ours)	98.01±.03	67.41±.07	44.69±.15	88.87±.01	61.75±.33	86.42±.35
TextCNN <sub>tune</sub> (Ours)	98.49±.06	69.26±.31	46.85±.18	90.06±.35	63.24±.34	86.11±1.17
9-VDCNN(Reported)	98.75	-	-	90.17	61.96	-
9-VDCNN(Github)	98.35	-	-	89.22	61.18	-
9-VDCNN(Word-level)	98.34	69.24	46.15	89.50	64.65	81.38
9-VDCNN +GraVeR	98.51	73.34	49.58	91.14	64.65	87.66
24-VDCNN(Word-level)	98.34	55.63	41.11	80.20	61.01	51.02

However, our framework is fully based on words, whereas the model is based on character-embedding. So, we slightly modified the model to use word embeddings and use the same pre-processing codes. Our modification does not harm original performance that much, as seen in Table 10.

We also attempt to use 29-depth, which showed the best performance in their paper. The results in DBpedia and Yelp are almost the same with Depth=9, but it overfits to small datasets such as AGNews, Yahoo, and IMDB.

In conclusion, one of the strong baseline, Very Deep Convolutional Neural Net is ‘slightly’ better, and GraVeR also works in the model. Therefore, we claim that our TextCNN<sub>tune</sub> is not a weak baseline.

## APPENDIX D. WORD DISTRIBUTION VISUALIZATION &amp; OTHER CUE WORDS

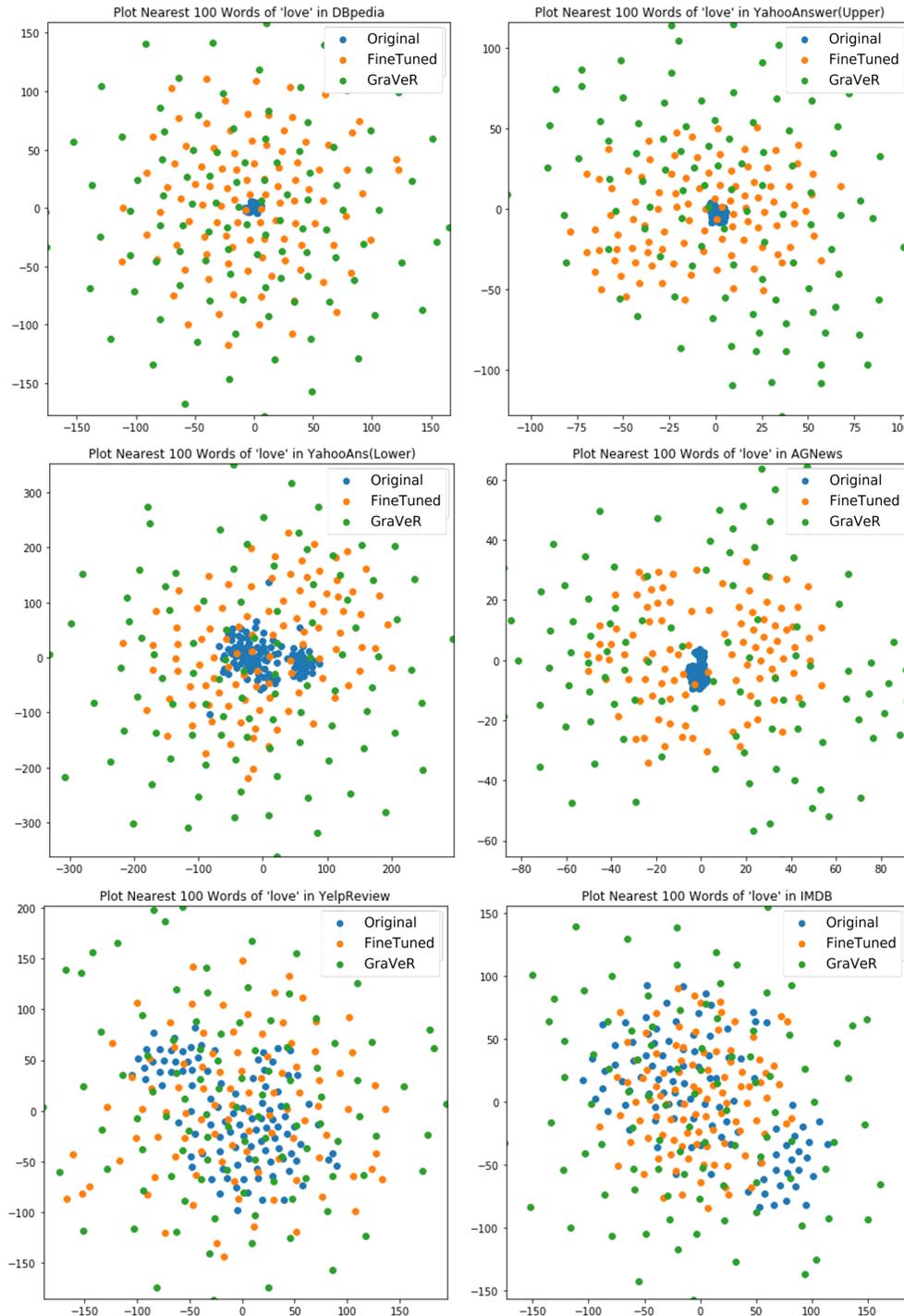


Figure 3: Plots of nearest top-100 words of a cue word (love) in initial embedding (Initial), after fine-tuned once (FineTuned), and our method GraVeR in 6 text classification tasks. The word vector distribution is largely changed through GraVeR when compared with the fine-tuned once, which is extracted from 1 conventional training. Note that the distribution of GraVeR representation is trained further than the fine-tuned once embedding.

Table 11: List of top-20 nearest words of cue words in initial embedding, after fine-tuned once, and our method GraVeR. The difference between initial embedding and fine-tuned once, and fine-tuned once and GraVeR are marked in bold and underlined, respectively. GraVeR attempts to find better embedding distributions.

Word	Method	Top-20 Nearest Words
hate	Initial	dont(.73),stupid(.72),hates(.72),think(.71),why(.69),love(.69),hating(.69),hated(.69),shit(.68),know(.68),damn(.68),say(.68),crap(.67),believe(.67),n't(.67),want(.67),saying(.67),dislike(.66),thing(.66),because(.66)
	Fine Tuned	dont(.72),hates(.71),stupid(.7),hating(.67),shit(.66),believe(.66),think(.66),hated(.65),crap(.65),know(.65), <u>afraid(.64)</u> ,love(.64), <u>hatred(.64)</u> , <u>cant(.64)</u> ,because(.63), <u>cuz(.63)</u> ,dislike(.63),damn(.63), <u>fear(.63)</u> ,want(.63)
	GraVeR	hates(.7),stupid(.67),hating(.66),think(.65),know(.65),believe(.64),shit(.64),crap(.64), <u>afraid(.63)</u> ,because(.63),want(.63),dislike(.63), <u>hatred(.63)</u> ,hated(.62), <u>ppl(.62)</u> ,damn(.62), <u>cuz(.61)</u> , <u>blame(.61)</u> , <u>realize(.61)</u> , <u>honestly(.61)</u>
peace	Initial	peaceful(.64),freedom(.63),hope(.63),unity(.61),happiness(.6),harmony(.59),prosperity(.58),prayer(.57),democracy(.57),faith(.57),conflict(.57),god(.56),bring(.56),justice(.56),friendship(.56),life(.56),love(.55),joy(.54),truth(.54),war(.54),
	Fine Tuned	peaceful(.61),justice(.57),freedom(.56),harmony(.56),happiness(.54),friendship(.54),hope(.53),bring(.53), <u>wish(.53)</u> ,unity(.53),god(.52),war(.52),democracy(.52),prosperity(.52),conflict(.51), <u>promise(.51)</u> , <u>_(.51)</u> ,faith(.51),cooperation(.51), <u>calm(.51)</u>
	GraVeR	peaceful(.61),harmony(.55),justice(.55),happiness(.54),unity(.54),freedom(.53),friendship(.52),god(.52), <u>wish(.51)</u> , <u>him(.5)</u> ,war(.5), <u>tranquility(.5)</u> , <u>cooperation(.5)</u> ,prosperity(.5), <u>calm(.5)</u> , <u>independence(.49)</u> ,democracy(.49), <u>rest(.49)</u> ,hope(.49), <u>pray(.49)</u>